UNITED STATES PATENT APPLICATION

OF

OMAR C. BALDONADO

SEAN P. FINN

MANSOUR J. KARAM

MICHAEL A. LLOYD

HERBERT S. MADAN

JAMES G. MCGUIRE

JOSE-MIGUEL PULIDO VILLAVERDE

FOR

METHOD AND APPARATUS FOR PERFORMANCE AND COST OPTIMIZATION IN AN INTERNETWORK

PREPARED BY WILSON SONSINI GOODRICH & ROSATI

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Applications No. 60/241,450, filed October 17, 2000 and 60/275,206 filed March 12, 2001, and U.S. utility applications 09/903,441, filed July 10, 2001 and 09/903,423, filed July 10, 2001 which are all hereby incorporated by reference in their entirety.

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to the field of networking. In particular, the invention relates to prioritizing and queuing updated routing information.

Description of the Related Art

Internetworks such as the Internet are currently comprised of Autonomous Systems, which exchange routing information via exterior gateway protocols. Amongst the most important of these protocols is the Border Gateway Protocol, or BGP. BGPv4 constructs a directed graph of the Autonomous Systems, based on the information exchanged between BGP routers. Each Autonomous System is identified by a unique 16 bit AS number, and, by use of the directed graphs, BGP ensures loop-free routing amongst the Autonomous Systems; BGP also enables the exchange of additional routing information between Autonomous Systems. BGP is further described in several RFCs, which are compiled in The Big Book of Border Gateway Protocol RFCs, by Pete Loshin, which is hereby incorporated by reference.

The Border Gateway Protocol provides network administrators some measure of control over outbound traffic control from their respective organizations. For instance, the protocol includes a LOCAL_PREF attribute, which allows BGP speakers to inform

other BGP speakers within the Autonomous System of the speaker's preference for an advertised route. The local preference attribute includes a degree of preference for the advertised route, which enables comparison against other routes for the same destination. As the LOCAL_PREF attribute is shared with other routers within an Autonomous System via IBGP, it determines outbound routes used by routers within the Autonomous System.

A WEIGHT parameter may also be used to indicate route preferences; higher preferences are assigned to routes with higher values of WEIGHT. The WEIGHT parameter is a proprietary addition to the BGPv4 supported by Cisco Systems, Inc. of San Jose, CA. In typical implementations, the WEIGHT parameter is given higher precedence than other BGP attributes.

The performance knobs described above are, however, rather simple, as they do not offer system administrators with sufficiently sophisticated means for enabling routers to discriminate amongst routes. There is a need for technology that enables greater control over outbound routing policy. In particular, there is a need to allow performance data about routes to be exchanged between routers. Additionally, system administrators should be able to fine tune routing policy based upon sophisticated, up-to-date measurements of route performance and pricing analysis of various routes.

SUMMARY OF THE INVENTION

The invention includes routing intelligence for evaluating routing paths based on performance measurements. In some embodiments of the invention, the routing intelligence may include processes executed in a self-contained device. This device may control one or more edge routers, based on performance data from end users. In other embodiments of the invention, the routing intelligence device may be used solely to

monitor one or more edge routers, producing reports but not effecting any changes to routing. Routing decisions may be injected to the edge routers via BGP updates. The devices may be stationed at the premises of a multihomed organization, such as an enterprise, ISP, government organization, university, or other organization supporting a sub-network coupled to an internetwork. In other embodiments, the routing intelligence comprises processes executed on a router.

In some embodiments, the routing intelligence unit may be a self-contained device controlling a single edge router. In other embodiments, a single routing intelligence unit controls multiple edge routers. Though the collection of routers s coupled to one or more Internet Service Provider (ISP) links, the individual routers may be coupled to one or more ISP links, or to no ISP links.

In some embodiments of the invention, the routing intelligence unit includes a main memory database, for storing information on network prefixes. In some embodiments, a plurality of geographically dispersed routing intelligence devices are coupled to a Routing Intelligence Exchange (RIX), which transmits performance data for network prefixes between the routing intelligence devices. These and other embodiments are described further herein.

BRIEF DESCRIPTION OF THE FIGURES

Fig. 1 – Fig.4 illustrate different configurations of routing intelligence units and edge routers, according to some embodiments of the invention.

Figure 5 schematically illustrates an internal architecture of a routing intelligence unit according to some embodiments of the invention.

Figure 6 illustrates a queuing and threading structure used in the routing intelligence unit in some embodiments of the invention.

DETAILED DESCRIPTION

A. System Overview

In some embodiments of the invention, one or more routing intelligence units are stationed at the premises of a multi-homed organization, each of which controls one or more edge routers. These devices inject BGP updates to the Edge Routers they control, based on performance data from measurements obtained locally, or from a Routing Intelligence Exchange—Routing Intelligence Exchanges are further described in U.S. Applications 60/241,450, 60/275,206, 09/903,441, and 09/903,423 which are hereby incorporated by reference in their entirety. Different configurations of these routing intelligence units and edge routers are illustrated in Figures 1 through 4. In some embodiments illustrated in Figure 1, one edge router 102 with multiple ISPs 104 and 106 is controlled by a single device 100. Figure 2 illustrates embodiments in which the routing intelligence unit 200 controls multiple edge routers 202 and 204, each of which in turn links to multiple ISPs 206, 208, 210, and 212; Figure 2 also illustrates embodiments in which routers 203 205 controlled by the routing intelligence unit 200 are not coupled to SPALs. In Figure 3, a single routing intelligence unit 300 controls multiple edge routers 302 and 304, each of which is linked to exactly one ISP 306 and 308. In additional embodiments illustrated in Figure 4, different routing intelligence units 400 and 402, each connected to a set of local edge routers 404, 406, 408, and 410, may coordinate their decisions. In some embodiments of the invention, the routing intelligence units comprise processes running within one or more processors housed in the edge routers. Other configurations of routing intelligence units and edge routers will be apparent to those skilled in the art.

B. Architecture of Routing Intelligence Units

The routing intelligence units include a Decision Maker resource. At a high level, the objective of the Decision Maker is to improve the end-user, application level performance of prefixes whenever the differential in performance between the best route and the default BGP route is significant. This general objective has two aspects:

- One goal is to reach a steady state whereby prefixes are, most of the time, routed through the best available Service Provider Access Link (i.e., SPAL), that is, through the SPAL that is the best in terms of end-to-end user performance for users belonging to the address space corresponding to that prefix. To achieve this goal, the Decision Maker will send a significant amount of updates to the router (over a tunable period of time) until steady state is reached. This desirable steady state results from a mix of customer-tunable criteria, which may include but are not limited to end-to-end user measurements, load on the links, and/or cost of the links.
- Current measurements of end-to-end user performance on the Internet show that fluctuations in performance are frequent. Indeed, the reasons for deterioration of performance of a prefix may include, but are not limited to the following:

  The network conditions can vary along the path used by the packets that correspond to that prefix on their way to their destination.

  Alternatively, the access link through which the prefix is routed can go down.

  The Service Provider to which the prefix is routed can lose coverage for that prefix.

In such occurrences, the routing intelligence unit should detect the deterioration/failure, and quickly take action to alleviate its effect on the end-user.

In order to optimize application performance, the routing intelligence unit converts measurements on the performance of routes traversing the edge-routers into scores that rate the quality of the end-to-end user experience. This score depends on the application of interest, namely voice, video and HTTP web traffic. In some embodiments of the invention, by default, the routing intelligence unit attempts to optimize the performance of web applications, so its decisions are based on a score model for HTTP. However, in such embodiments, the customer has the choice between all of voice, video, and HTTP.

In order to avoid swamping routers with BGP updates, in some embodiments of the invention, the maximum rate of update permitted by the prefix scheduler is offered as, for example, a control, such as a knob that is set by the customer. The faster the rate of updates, the faster the system can react in the event of specific performance deteriorations or link failures.

However, the rate of updates should be low enough not to overwhelm the router. In some embodiments, the selected rate will depend on the customer's setting (e.g., the traffic pattern, link bandwidth, etc.); for example, faster rates are reserved to large enterprises where the number of covered prefixes is large. Even when the rate of updates is slow, in some embodiments of the invention, the most urgent updates are still scheduled first: this is performed by sorting the prefix update requests in a priority queue as a function of their urgency. The priority queue is then maintained in priority order. The most urgent events (such as loss of coverage, or link failure) bypass this queue and are dealt with immediately.

In case interface statistics are available, the Decision Maker may directly use the corresponding information to function in an optimized way. For example, in some embodiments of the invention, the Decision Maker can use bandwidth information to make sure that a link of lower bandwidth is not swamped by too much traffic; in a similar

manner, link utilization can be used to affect the rate of BGP updates sent to the router. Finally, the prefix scheduler may use per-link cost information, as provided by the user to tailor its operation. For example, assume that the router is connected to the Internet through two links: Link 1 is a full T3, while Link 2 is a burstable T3, limited to 3 Mbit/sec. That is, whenever load exceeds the 3 Mbit/sec mark on Link 2, the user incurs a penalty cost. Combining information pertaining to per-link cost and utilization, the Decision Maker can attempt to minimize the instances in which load exceeds 3 Mbit/sec on Link 2, thus resulting in reduced costs to the user.

In some implementations, the Decision Maker may also use configurable preference weights to adjust link selection. The cost of carrying traffic may vary between links, or a user may for other reasons prefer the use of certain links. The Decision Maker can attempt to direct traffic away from some links and towards others by penalizing the measurements obtained on the less preferred links; this encourages use of the preferred links, but still allows the less preferred links to carry any traffic which receives great benefit.

Even though information about SPALs (e.g., the bandwidth and utilization of each of the access links) and prefixes (e.g., the load profile of a particular prefix) is valuable and can be used effectively (as described above) to achieve a more optimal results (according to the algorithm's objective), the Decision Maker is designed to work well even if the only available information is provided by edge stats measurements.

In case the routing intelligence unit fails, the design is such that the edge router falls back to the routing that is specified in the BGP feed. The same behavior occurs in case performance routes fail. Finally, in some embodiments of the invention, a flapping control algorithm is included in the design, avoiding the occurrence of undesirable excessive flapping of a prefix among the different access links.

A diagram showing the high-level architecture of Routing Intelligence Unit, and focused on its BGP settings is shown in Figure 5. In the embodiments illustrated in Figure 5, three BGP peering types may exist between a given Routing Intelligence Unit 500 and the external world: one to control the local edge router or routers 502 that this particular Routing Intelligence Unit 500 is optimizing, one to a Routing Infrastructure Exchange (RIX) 504, and one to every other Routing Intelligence Unit device with which it coordinates 506, as further described in U.S. applications 60/241,450, 60/275,206, 09/903,441, and 09/903,423, which are hereby incorporated by reference in their entirety. In the diagram shown in Figure 5, the three external peering types are shown as the arrows at far left (to the Edge Routers 502 and to RIX 504) and far right 506. In order for BGP updates to be propagated to the appropriate devices, some devices are configured to be route reflectors, and others as route reflector clients. In some embodiments of the invention, the Decision Maker is a reflector client on all its iBGP peering types.

C. Queuing Architecture

A diagram showing the high level mechanics of the decision maker prefix scheduler is shown in Figure 6. As illustrated in Figure 6, two threads essentially drive the operation of the scheduler. The first thread polls the database for changes in terms of per-SPAL performance, load, or coverage, and decides on which prefix updates to insert in a Priority Queue that holds prefix update requests. The second thread takes items out of the queue in a rate-controlled fashion, and converts the corresponding update requests into an appropriate set of NLRIs (Network Layer Reachability Information) that it sends to the local routers, and an appropriate set of NLRIs that it sends to the back channel for communication to other Routing Intelligence Units.

In the following, we describe each thread separately. In the description, we will refer to tables in the database, and to fields within these tables. The contents of this database are also explicated in U.S. Applications 60/241,450, 60/275,206, 09/903,441, 09/903,423 which are hereby incorporated by reference in their entirety.

Thread 1

This first thread 600 polls the database for changes in terms of per-SPAL performance, load, or coverage, and decides on which prefix updates to insert in a Priority Queue that holds prefix update requests.

In some embodiments of the invention, such changes are checked for in 2 passes. The first pass looks for group level changes, wherein a group comprises an arbitrary collection of prefixes. Groups are also described in U.S. Applications 60/241,450, 60/275,206, 09/903,441, 09/903,423 which are hereby incorporated by reference in their entirety. In case a significant change in performance for a group is noticed, the group is unpacked into its individual prefixes; the corresponding prefixes are checked and considered for insertion in the priority queue. The second pass captures prefixes for which there are no group-level performance changes.

The circumstances under which an update request for a prefix is made may include any one or more of the following:

1. In case a significant change in its performance score is witnessed on at least one of its local SPALs.
2. In case a significant change in its performance score is witnessed on a foreign SPAL (that is, a SPAL that is controlled by a different Routing Intelligence Unit box in a coordinated system).
3. In case any of the local SPALs becomes invalid.

4. In case an update pertaining to this prefix was received from the router.

Note that measurements reside at the group level; hence, Check 1 can be done in the first pass. On the other hand, all of Checks 2, 3, and 4 are prefix-specific and may be performed in Pass 2: indeed, foreign performance updates are transferred through the back channel in BGP messages, and hence correspond to particular prefixes. Also, SPALs may become invalid for some, and not necessary all prefixes in a group. Finally, updates from the router relate to the change of winner SPALs for some prefixes, or to the withdrawal of other prefixes. (In fact, any information that is transferred by BGP relates to prefixes.)

Pass 1:

In some embodiments of the invention, in the first pass, an asynchronous thread goes through all groups in the GROUP_SPAL table, checking whether the NEW_DATA bit is set. This bit is set by the measurement listener in case a new measurement from a /32 resulted in an update of delay, jitter, and loss in the database. Delay, jitter, and loss, also denoted as d, v, and p, are used to compute an application-specific score, denoted by **m**. The scalar **m**. is used to rate application-specific performance; MOS stands for "Mean Opinion Score", and represents the synthetic application-specific performance. In embodiments of the invention, MOS may be multiplied by a degradation factor that is function of link utilization, resulting in **m**. (That is, the larger the utilization of a given SPAL, the larger the degradation factor, and the lower the resulting **m**)

In embodiments of the invention, users of the device may also configure penalty factors per SPAL. Non-limiting examples of the uses of such penalty features include handicapping some links relative to others, to achieving cost control, or accomplishing

other policy objectives. As a non-limiting example, Provider X may charge substantially more per unit of bandwidth than Provider Y. In such a situation, the penalty feature allows the user to apply an **m** penalty to SPAL X. This will cause Provider Y to receive more traffic, except for those prefixes in which the performance of Provider X is substantially better. One implementation of this embodiment is to subtract the penalty for the appropriate SPAL after **m** is computed. Other implementations of the penalty feature will be apparent to those skilled in the art.

Even when NEW_DATA is set, the variation in d, v, and p can be small enough so that the change in the resulting scalar **m** is insignificant. Hence, in some embodiments of the invention, the prefix is only considered for insertion in the queue in case the change in **m** is significant enough. The corresponding pseudo-code is shown below.

```
for each group
{
                // First pass: only consider groups for which there
is a change in the group pref data
        compute_winner_set = 0;

        for each spal (<> other)
        {
                // check whether there is new data for this group
                if (new_data(group, spal)==1)
                {
                        compute m (spal, d, v, p, spal-penalty),
store in local memory
```

```
                        new_data(group, spal) = 0;
                        if (significant change in m)
                        {
                                store m (spal, d, v, p) in
group_spal
                                compute_winner_set = 1;
                                break;
                        }
                }
        }

        if (compute_winner_set)
                for each prefix
                        schedule_prefix(prefix) // see below
```

In some embodiments of the invention, rolling averages are used to update measurements of delay, jitter, and loss, i.e.,

d = alpha*d + (1 – alpha)*dnew

v = beta*v + (1 – beta)*vnew

p = gamma*p + (1 – gamma)*pnew,

where dnew, vnew, pnew represent the new delay, jitter, and loss measurements. Algorithms for calculating MOS for HTTP (1.0 and 1.1) and for voice and video are also presented in U.S. Provisional Application 60/241,450, filed October 17, 2000 and 60/275,206 filed March 12, 2001. Values used for the models employed by these algorithms in embodiments of the invention are presented in an XML format below. Note that since MOS is computed per group, a selection from the sets of the following parameters may be made to allow different optimization goals for each group.

<module> <engine slot="1"> <application model="http1.0" [alpha="0.9" beta="0.9" gamma="0.9" theta="1.18" phi="0.13" omega="0.15" psi="0.25"] />
</engine> </module>

<module> <engine slot="1"> <application model="http1.1" [alpha="0.9" beta="0.9" gamma="0.9" theta="1.3" phi="0.31" omega="0.41" psi="1.0"] />
</engine> </module>

<module> <engine slot="1"> <application model="voice" [alpha="0.9" beta="0.9" gamma="0.9" theta ="1.5" phi="6.0" omega="23.0" psi="0.0"] /> </engine>
</module>
<module> <engine slot="1"> <application model="video" [alpha="0.9" beta="0.9" gamma="0.9" theta="1.0" phi="4.0" omega="69.0" psi="0.0"] />
</engine> </module>

The values presented above are given as examples only. Many different models for deriving MOS scores for different applications will be apparent to those skilled in the art.

Pass 2

In the second pass, an asynchronous thread goes through all prefixes in the PREFIX table. For each prefix, Checks 2, 3, and 4 are made: NEW_INCOMING_BID in the PREFIX table indicates that a new bid was received from the coordination back channel; NEW_INVALID in the PREFIX_SPAL table indicates, for a particular (Prefix P,

SPAL x) pair a loss of coverage for Prefix P over SPAL x. NEW_NATURAL_DATA indicates the receipt by Routing Intelligence Unit of an update message from a router, notifying it of a change in its natural BGP winner. In fact, the Decision Maker only asserts a performance route in case it is not the same as the natural BGP route; hence, it can potentially receive updates concerning the natural BGP winners of given prefixes from routers to which it has asserted no performance route for those prefixes. (If Routing Intelligence Unit were to assert performance routes regarding a given prefix P to all routers irrespectively of the current BGP winner for that prefix, it will never receive an update from the router pertaining to changes in the natural BGP winner for Prefix P. Indeed, the performance route would always be the winner, so the router would assume there is nothing to talk about.)

The following example illustrates the usefulness of the NEW_NATURAL_DATA flag: Assume that the Decision Maker controls 3 routers, each of which controls its individual SPAL. Assume that the Decision Maker has just determined that Prefix P will move to SPAL 1. Assume that Prefix P believes that the natural BGP route for Prefix P as saved by Router 1 is SPAL 1, the same as its current performance assertion. The Decision Maker's logical operation is to withdraw Prefix P's last performance route (say SPAL 3). However, it turned out that this BGP natural route has, in fact changed to SPAL 2; indeed, this could have happened during the previous assertion of a performance route for Prefix P (since, in this case, as mentioned above, the Decision Maker receives no updates for Prefix P from the router, despite potential changes in Prefix P's natural BGP winner). As a result of this discrepancy, all traffic pertaining to Prefix P will be routed through SPAL 2, the current natural BGP winner for Prefix P, which is not the desired behavior.

This is the primary reason for NEW_NATURAL_DATA: as such an event occurs, the router sends an update back to the Decision Maker, communicating to it the change in

natural route. The Peer Manager sees the change in natural BGP route and sets the NEW_NATURAL_DATA flag to 1; consequently, the prefix is considered for re-scheduling during this pass, in Thread 1, as described above. Note that in case of changes in the natural BGP route for a given prefix, the Decision Maker will need two passes through the Priority Queue before the prefix is routed through its appropriate performance route.

Finally, the ACCEPTING_DATA bit in the prefix table is checked. ACCEPTING_DATA is set to 0 by the peer manager to notify the decision maker not to assert performance routes for this prefix. This would primarily occur in case the prefix is withdrawn from the BGP tables in all local routers. In this case, in the ROUTER_PREFIX_SPAL table, the ANNOUNCED bit would be set to 0 on all routers and all SPALs for that prefix. Clearly, a prefix is only considered for insertion in the queue in case ACCEPTING_DATA is set to 1.

```
for each prefix
{
        //Checks 2 and 4: scan the prefix_group table
        get new_bid, new_natural, and accepting_data from
prefix_group
        if (new_bid)||(new_natural)
        {
                if (accepting_data)
                {
                        schedule_prefix(prefix) // see below
                }
        }
```

```
        //Check 3: scan the prefix_spal table
        get new_invalid, from prefix_spal
        if (new_invalid)
        {
                schedule_prefix(prefix)          }
}
```

Note that asserting a performance route about a prefix that does not exist in any of the routers' BGP tables could be problematic, depending on the surrounding network environment. If the set of controlled routers do not emit routes to any other BGP routers, then it is acceptable to generate new prefixes. But if any propagation is possible, there is a danger of generating an attractor for some traffic.

Specifically, if the new route is the most specific route known for some addresses, then any traffic to those addresses will tend to forward from uncontrolled routers towards the controlled routers. This can be very disruptive, since such routing decisions could be very far from optimal.

The mechanism can cope with this in a number of ways:

- Prevent any use of a prefix unknown to BGP. This is achieved using the ACCEPTING_DATA check included in some embodiments of the invention.
- Permit all such use, in a context where new routes cannot propagate
- Permit such use, but mark any new prefix with the well-known community value no-advertise to prevent propagation
- Permit such use, but configure the routers to prevent any further propagation (in some embodiments, by filtering such prefixes)

Deciding to Insert a Prefix Update Request in the Priority Queue: The *schedule_prefix* Function

Once a prefix P makes it through the checks imposed in either Pass 1 or Pass 2, it is considered for insertion into the prefix update priority queue. `schedule_prefix` includes the related functionality, described below:

- First of all, a winner set of SPALs is re-computed for P; this set includes SPALs for which the performance is close to maximal.
- After the winner set W is computed for P, the decision maker determines whether the current route for P is included in W.
- In case of a coordinated Routing Intelligence Unit system, in some embodiments of the invention, the back channel is sent updates pertaining to Prefix P even if the local prefix update request is dropped. For example, the performance on local links could have changed dramatically since the last time a bid was sent to the back channel for this prefix; in the event of such an occurrence, an updated bid is sent to the back channel (through the BGP peering set up for this purpose).
- In case the current route is not part of the newly computed winner set, it is clear that Prefix P is not routed optimally. Before going ahead and inserting an update request for Prefix P in the queue, the Routing Intelligence Unit performs a check of the flapping history for Prefix P. In case this check shows that Prefix P has an excessive tendency to flap, no prefix update request is inserted in the queue.
- In some embodiments of the invention, before the prefix is inserted in the queue, a SPAL is chosen at random from the winner set. In case the winner set includes a remote SPAL controlled by a coordinated Routing Intelligence Unit as well as a local SPAL, the local SPAL is always preferred. Also, in some embodiments of the

invention, the randomness may be tweaked according to factors pertaining to any one or more of the following: link bandwidth, link cost, and traffic load for a given prefix. Finally, the state in the database is updated, and the element is inserted in the Priority Queue. The rank of the prefix update in the priority queue is determined by computing the potential percent improvement obtained from moving the prefix from its current route to the pending winner route.

At the outset, a winner set of SPALs is re-computed for P; this set includes SPALs for which the performance is close to maximal. In some embodiments of the invention, invalid SPALs are excluded from the winner set computation. Bids from remote SPALs under the control of coordinated Routing Intelligence Units may, in embodiments, be included in the winner set computation. Since the bids corresponding to such remote routes are filtered through BGP, they are in units which are compatible with iBGP's LocalPref, which in some implementations is limited to 0-255. Therefore one possible implementation is to multiply **m** by 255. The converted quantity is referred to as MSLP. For consistency, the **m** values computed for local SPALs are also converted to local_pref units. The new winner is then determined to be the set of all SPALs for which MSLP is larger than MSLPmax - winner-set-threshold, where $MSLP_{max}$ represents the maximum MSLP for that prefix across all available SPALs, and winner-set-threshold represents a customer-tunable threshold threshold specified in LocalPref units. The related pseudo-code is shown below.

```
for each spal (<> other)
{
        get invalid bit from prefix_spal
```

```
        if (invalid)
        {
                mark spal as invalid, not to be used in winner_set
computation
                continue
        }
        convert m (spal) to MSLP
        Store MSLP in prefix_spal table
}
for spal=other
{
        get MSLP_other = other_bid in prefix_group table
}
compute winner_set(prefix) // considers winners among all valid
spals and other_bid
```

After the winner set W is computed for P, the decision maker determines whether the current route for P is included in W. Indeed, in such a case, the performance of that prefix can't be improved much further, so no prefix update request needs to be inserted in the queue.

Even though an update request for a given prefix is ignored, the Decision Maker may still send an update to the back channel in certain embodiments. For example, even though the current route for Prefix P is still part of the winner set, performance degradation could have affected all SPALs at once, in which case the bid that was previously sent to the back channel for Prefix P is probably inaccurate. In some embodiments, one may solve this problem by implementing the following: the last bid for

a given prefix is saved as MY_BID in the PREFIX table; a low and high threshold are then computed using two user-configurable parameters, `bid-threshold-low` and `bid-threshold-high`. In case of a significant difference between the MSLP score on the current route and the last score sent to the back channel for that prefix (i.e., MY_BID) is witnessed (that is, if the new score falls below (1-`bid-threshold-low`)*100% or jumps to a value that is larger than (1+`bid-threshold-high`)*100% of MY_BID), a BGP message is sent to the back channel, carrying the new bid for Prefix P to remote coordinated Routing Intelligence Units. Pseudo-code illustrating the functionality described here is shown below.

```
//First, detect non-communicated withdrawal of a prefix
if winner_set only comprises remote link
{
        for all local routers
                if performance route exists for that (prefix,
router) pair in the ROUTER_PREFIX_SPAL table
                        send urgent withdrawal of this route to
edge router
        continue
}
get current_winner(prefix) and pending_winner(prefix) from
prefix_spal table

if (pending_winner!=current_winner)
{
        if (current_winner in winner_set)
```

```
        {
                update pending_winner = current_winner in database
                continue
        }
        if (current_winner not in winner_set)&&(pending_winner in
winner_set)
        {
                continue
        }
        //if (current_winner not in winner_set)&&(pending_winner
not in winner_set)
        //{
        //}
}

if (current_winner==pending_winner)
{
        if (new_natural)
        {
                for all routers
                {
                        current_route_per_router = SPAL(prefix,
router, type = natural, state = latest_ON)
                        if (current_route_per_router exists) &&
(current_route_per_router != current_winner)
                        {
```

```
                                        special_route =
current_route_per_router
                                        set local special_route_flag = 1;
                                        break;
                                }
                        }
                }
                else
                {
                        current_route = current_winner
                }
                if (current_route in
winner_set)||(special_route==current_winner)
                {
                        get bid_low_threshold and bid_high_threshold from
prefix_group table
                        if ((MSLP(prefix, current_spal) <
bid_low_threshold)||(MSLP(prefix, current_spal)
bid_high_threshold))
                        {
                                compute bid_low_threshold and
bid_high_threshold from MSLP(prefix)
                                store bid_low_threshold and
bid_high_threshold in prefix_group
                                form NLRI to send to backchannel SBGP
                        }
                        continue
```

```
        }
}
```

At this point, it is clear that Prefix P is not routed optimally. In some embodiments of the invention, before proceeding with sending the update request to the edge router, the Routing Intelligence Unit performs a check of the flapping history for Prefix P. An algorithm whose operation is very close to the flapping detection algorithm in BGP monitors the flapping history of a prefix. The algorithm can be controlled by, in one embodiment, three user-controlled parameters `flap_weight`, `flap_low`, and `flap_high` and works as follows: the tendency of a prefix to flap is monitored by a variable denoted FORGIVING_MODE that resides in the PREFIX table. FORGIVING_MODE and other flapping parameters are updated in Thread 2 right before a performance route pertaining to Prefix P is asserted to the local routers. In case FORGIVING_MODE is set to 1, the tendency for Prefix P to flap is considered excessive, and the prefix update request is ignored. Conversely, in case FORGIVING_MODE is set to 0, Prefix P has no abnormal tendency to flap, so it is safe to consider its update request.

```
get flapping state for prefix from prefix_group table
if (excessive flapping)
{
        continue
}
```

If a prefix survives to this point in Thread 1, it will deterministically be inserted in the queue. Hence, all bits that were checked should be reset at this point so that some other pass on the prefixes does not reconsider and reschedule the prefix update request. For

example, in case the prefix belongs to a group for which there was a significant change in **m**, the prefix will be considered for insertion in the queue in Pass 1, and should not be reconsidered in Pass2.

```
//reset prefix level bits, if necessary
for each spal (<> other)
{
        get new_invalid bit from prefix_spal
        if (new_invalid)
        reset new_invalid to 0 in prefix_spal
}
get new_bid and new_natural bits from prefix_group
if (new_bid)
        reset new_bid to 0 in prefix_group
if (new_natural)
        reset new_natural to 0 in prefix_group
```

In some embodiments of the invention, before the prefix is inserted in the queue, a SPAL is chosen at random from the winner set. This way, traffic is spread across more than one SPAL, hence achieving some level of load balancing. In order to achieve some set of desirable policies, randomness can be tweaked in order to favor some SPALs and disregard others. For example, in case the winner set includes a remote SPAL controlled by a coordinated Routing Intelligence Unit as well as a local SPAL, the local SPAL is always preferred. In other words, a remote SPAL is only the winner in case it is the only available SPAL in the winner set. Also, depending on the weight of a prefix and the observed load on different links, one can tweak the probabilities in such a way that the

prefix is routed through a SPAL that fits it best. (This feature corresponds to the "Saturation Avoidance Factor" – SAF, described later in this document) After a winner is selected, PENDING_WINNER in PREFIX_SPAL is updated to reflect the new potential winner. Finally, the element is inserted in the Priority Queue. The rank of the prefix update in the priority queue is determined by computing the percent improvement; that is, the percent improvement obtained from moving the prefix from its current route to the pending winner route. That is, percent-improvement = [score(pending_winner) – Score(current_route)]/Score(current_route). The special-spal-flag is part of the data structure for the update, as it will be used in the determination of which messages to send to the local routers.

```
if ((winner_set_size>1) and (other in winner_set))
        remove other from winner_set
select spal from winner_set at random
update PENDING_WINNER in PREFIX_SPAL table
compute percent_improvement for prefix
insert prefix in prefix update queue
```

Thread 2

In this thread 602, elements are taken out of the queue in a rate-controlled manner. In some embodiments of the invention, this rate is specified by the customer. The update rate is often referred to as the token rate. Tokens are given at regular intervals, according to the update rate. Each time a token appears, the head of the queue is taken out of the queue, and considered for potential update. In case the database shows that more recent

passes in Thread 1 have canceled the update request, it is dropped *without losing the corresponding token*; the next update request is then taken out from the head of the queue; this procedure is performed until either the queue empties, or a valid request is obtained. In some embodiments of the invention, when an update request that corresponds to Prefix P is determined to be current (thus, valid), one or more of the following tasks are performed:

- The flapping state is updated for Prefix P.
- The database is updated to reflect the new actual winner; more specifically, the pending winner, chosen before inserting the prefix update request at the end of the first thread now becomes the current winner.
- The database is checked to determine the current state of each of the individual routers. Accordingly, individual NLRIs are formed and sent to each of the routers. For example, no performance route is sent to an edge router in case the BGP winner for Prefix P, according to that router is found to be the same.
- An NLRI is sent to the back channel, describing the new local winner.
- Finally, the database is updated to keep track of the messages that were sent to each of the routers, as well as the expected resulting state of these routers.

In this thread 602, elements are just taken out from the queue in a rate-controlled manner, according to an update rate that may be set by the customer. The update rate is often referred to as the token rate: indeed, tokens are given at regular intervals, according to the update rate. Each time a token appears, the head of the queue is taken out, and considered for potential update.

Assume that the update request concerns Prefix P. The PREFIX_SPAL table is checked to obtain the PENDING_WINNER and CURRENT_WINNER for Prefix P.

In case PENDING_WINNER and CURRENT_WINNER correspond to the same SPAL, this is an indication that a more recent pass in Thread 1 has canceled the update request; in this case, the update request is dropped, *without losing the corresponding token*; the next token request is then polled from the head of the queue; this procedure is performed until either the queue empties, or a valid request, for which PENDING_WINNER and CURRENT_WINNER are different, is obtained.

Having different pending and current winners reflects a valid update request. In this case, the Decision Maker should assert the winning route for Prefix P; correspondingly, a series of tasks are performed. First, the flapping state is updated for Prefix P. In some embodiments of the invention, the tendency of a prefix to flap is monitored by a variable denoted INTERCHANGE_RATE that resides in the PREFIX table. The `flap_weight` parameter dictates the dynamics of INTERCHANGE_RATE; more specifically, at this point in the algorithm thread, INTERCHANGE_RATE is updated using the last value of INTERCHANGE_RATE, as stored in the table, LAST_ICR_TIME, also stored in the PREFIX table, and `flap_weight`. In case the new computed INTERCHANGE_RATE is below `flap_low`, Routing Intelligence Unit considers the tendency for that prefix to flap to be low. On the other hand, when INTERCHANGE_RATE exceeds `flap_high`, the Routing Intelligence Unit considers the tendency for that prefix to flap to be high. That is, the algorithm functions in the following fashion:

- In case FORGIVING_MODE (also in the PREFIX table) is set to 0, and INTERCHANGE_RATE exceeds `flap_high`, FORGIVING_MODE is set to 1.
- In case FORGIVING_MODE is set to 1, but INTERCHANGE_RATE drops below `flap_low`, FORGIVING_MODE is set to 0 again, and the prefix update request survives this check.

- In case FORGIVING_MODE is set to 1 and INTERCHANGE_RATE is larger than `flap_low`, or FORGIVING_MODE is set to 0, and INTERCHANGE_RATE is below `flap_high`, FORGIVING_MODE does not change.

Note that the method presented above is only one technique for controlling flapping; others will be apparent to those skilled in the art.

In some embodiments of the invention, the two parameters `flap_low`, and `flap_high` are separated by an amount to avoid hysterisis between the two values. Then, the Decision Maker updates the PREFIX_SPAL table to reflect this change; more specifically, CURRENT_WINNER is moved to PENDING_WINNER in the table. At this time, the ROUTER_PREFIX_SPAL table is queried to capture the current state of each router in regards to Prefix P. Accordingly, different NLRIs are formed and sent to each of the routers.

In some embodiments of the invention, the Decision Maker only asserts a performance route in case it is not the same as the natural BGP route; indeed, if Routing Intelligence Unit were to assert performance routes regarding a given prefix P to all routers irrespectively of the current BGP winner for that prefix, it will never receive an update from the router pertaining to changes in the natural BGP winner for Prefix P. (Indeed, the performance route would always be the winner, so the router would assume there is nothing to talk about.)

Also, an NLRI is sent to the back channel, describing to other Routing Intelligence Units in a coordinated system the new local winner. Finally, the database is updated to keep track of the messages that were sent to each of the routers, as well as the expected resulting state of these routers.

Prior to forming the NLRIs, the database is updated as to include the new flap parameters and prefix-SPAL information (i.e., the new current SPAL for that prefix). The

BGP update sent to an edge router may be filtered out by policy on the router. However, assuming the update is permissible, it may be made to win in the router's BGP comparison process. One implementation is to have the edge router to apply a high Weight value to the incoming update. (Weight is a common BGP knob, supported in most major implementations of the protocol, but it is not in the original protocol specification) This technique constrains the update so that it gains an advantage only on the router or routers to which the update is directly sent; this is desirable if some other routers are not controlled by a device such as the one described here. It is also possible to send the update with normal BGP attributes which make the route attractive, such as a high LocalPref value.

```
if (local_token available)
{
        get prefix at the head of the local update queue
        updatePrefixSpal(prefix, spal)
        updateFlapStats(prefix)
        compute bid_low_threshold and bid_high_threshold from
MSLP(prefix)
        store bid_low_threshold and bid_high_threshold in
prefix_group
        form NLRI to send to local SBGP
        form NLRI to send to backchannel SBGP
}
```

D. Technical Considerations

Queue Size

In some embodiments of the invention, a maximum queue size is to be chosen by the customer. In some embodiments, a small queue size may be chosen, so the maximum delay involved between the time instant a prefix update request is queued and the time instant it is considered by the second thread as a potential BGP update is small. For example, in case the token rate corresponding to a given link is 10 tokens per second, and we choose not to exceed a 2 second queuing delay, the queue should be able to accommodate 20 prefix update requests. Note that this method is simple, and only requires the knowledge of the token rate and the maximum acceptable delay.

Maximum Rate of Prefix Updates

It is desirable for the Routing Intelligence Unit to remain conservative in the rate of updates it communicates to the edge-router. This is the function of the token rate, which acts as a brake to the whole system. In some embodiments of the invention, the responsibility for setting the token rate is transferred to the customer, who selects a token rate that best fits her bandwidth and traffic pattern.

E. Feedback from the Listener BGP

The feedback from the listener BGP is valuable as it describes the actual current state of the local edge routers. Accordingly, in some embodiments of the invention, a separate routing intelligence unit thread modifies the content of the database according to the state it gets from the router(s). The Routing Intelligence Unit can operate more subtly in case it is a *perfect listener*; we consider the Routing Intelligence Unit to be a perfect listener if it has knowledge of the individual BGP feeds from each individual SPAL. That is, in case the Routing Intelligence Unit is connected to three access links, each connecting

to a separate provider, the Routing Intelligence Unit is a perfect listener if it has access to each of the three feeds handed by each of these providers.

Configuring Routing Intelligence Unit as a Perfect Listener is desirable, as it allows the support of private peerings. For example, unless Routing Intelligence Unit is configured as a Perfect listener, when Routing Intelligence Unit hears about a prefix, it can't assume that coverage exists for that prefix across all SPALs. Considering the scenario described above, a prefix that the Routing Intelligence Unit learns about could be covered by any of the three SPALs the router is connected to. For example, assume that only SPAL 1 has coverage for a given prefix P; in case the Routing Intelligence Unit asserts a performance route for that prefix across SPAL 2, there is no guarantee that the traffic pertaining to that prefix will be transited by the Service Provider to which SPAL 2 is connected (which we denote Provider 2). In case Provider 2 actually has a private peering with Provider X that obeys to some pre-specified contract, Provider X could well monitor the traffic from Provider 2, and filter all packets that do not conform to that contract. In case this contract namely specifies that Provider X will only provide transit to customers residing on Provider X's network, then the traffic pertaining to Prefix P will be dropped. If Routing Intelligence Unit were a Perfect Listener, it would only assert performance routes for prefixes across SPALs that are determined to have coverage for these prefixes. This behavior may be referred to as "extremely polite".

In some embodiments, the Routing Intelligence Unit is capable of avoiding the "Rocking the boat" problem, which stems from unwanted propagation of prefixes which did not already exist in BGP. The Routing Intelligence Unit can operate in "impolite" mode, where any prefixes may be used, or in "polite" mode, where only those prefixes which were previously present in BGP can be used. An ANNOUNCED bit resides in the ROUTER_PREFIX_SPAL table, and is set by the Peer Manager in case the Routing

Intelligence Unit hears about a prefix from any of the Routers. This bit allows use of "polite" mode by the following procedure: in case the ANNOUNCED bit is set to 0 for all (router, SPAL) combinations in the ROUTER_PREFIX_SPAL table, then ACCEPTING_DATA is set to 0 in the PREFIX table.

F. Urgent Events

In case a catastrophic event occurs, such as a link going down, some embodiments of the invention send urgent BGP updates to the router. These urgent updates have priority over the entire algorithm described above. For example, in case a SPAL has lost coverage for a prefix, an urgent BGP message should be sent to the router, requesting to move the prefix to other SPALs. A list of urgent events upon which such actions may be taken, and a description of the algorithms pertaining to these actions, are described below.

Algorithm for the Detection of an Invalid SPAL

In some embodiments of the invention, a specific (Prefix P, SPAL x) pair is invalidated in case there are reasons to believe that SPAL x no longer provides coverage to Prefix P. One possible implementation is described as follows. Measurements corresponding to a (Prefix, SPAL) pair are assumed to arrive to the Decision Maker at something close to a predictable rate. A background thread that is independent from Threads 1 and 2 computes this update rate, and stores a time of last update, the LAST_UPDATE_TIME. Another background thread verifies that LAST_ICR_TIME is reasonable given UPDATE_RATE. For example, assuming that measurements come in following a Poisson distribution, it is easy to verify whether LAST_ICR_TIME exceeds a fixed percentile of the inter-arrival interval. As LAST_UPDATE_TIME increases, the

Decision Maker becomes more and more worried about the validity of the path. In the current design, there are two thresholds: at the first threshold, the NEW_INVALID and INVALID flags are set in the PREFIX_SPAL table. As described in Thread 1 above, setting the NEW_INVALID flag for a (Prefix P, SPAL x) pair will prevent any new update requests for Prefix P to be routed through SPAL x. At this stage, no other action is taken. At the second threshold, the Decision Maker becomes "very concerned" about routing Prefix P through SPAL x; hence, an urgent check is made to see whether Prefix P is currently routed through SPAL x, in which case an urgent NLRI is created (that is, an NLRI that bypasses the entire queue system) in order to route Prefix through a different SPAL.

G. Saturation Avoidance Factor

Some embodiments of the invention support a Saturation Avoidance Factor, which measures the effect of a prefix on other prefixes. In some embodiments of the invention, the "Saturation Avoidance Factor" (SAF) pertaining to a given prefix may be taken into account when prefixes are sorted in the Priority Queue. This SAF measures the effect of a prefix on other prefixes. That is, if, upon scheduling a prefix on a given link, its effect on the other prefixes already scheduled on that link is high (i.e., this effectively means that the aggregate load for this prefix is large), its SAF should be low. The lower the SAF of a prefix, the lower its place in the Priority Queue. This way, the algorithm will always favor low load prefixes rather than high load prefixes. Note that in some embodiments, the SAF is not directly proportional to load. For example, a prefix that has a load equal to $0.75C$ has a different SAF whether it is considered to be scheduled on an empty link or on a link which utilization has already reached 75%. In the later case, the SAF should be as low as possible, since scheduling the prefix on the link would result in a link overflow.

At times, the token rate may be slower than the responded feedback. In case link utilization information is obtained through interface-stats, the token rate may be slower than the rate at which utilization information comes in. Also, the token rate may be slower than the rate at which edge-stats measurements come in.

Additionally, in some embodiments, each prefix is considered at a time. That is, PQServiceRate is small enough so that no more than one token is handed at a time. For example, denoting by T the token rate obtained from the above considerations, PQServiceRate is equal to $1/T$. If more than one token were handed at one time, two large prefixes could be scheduled on the same link, just as in the example above, potentially leading to bad performance.

In some embodiments of the invention, the SAF is a per-prefix, per-SPAL quantity. For example, assume that a prefix carries with it a load of 75% the capacity of all SPALs. If we have a choice between two SPALs, SPAL 1 and SPAL 2, SPAL 1 already carrying a load of 50 %, the other having a load of 0%. In this case, moving Prefix p to SPAL 1 will result in bad performance not only for itself, but also for all other prefixes already routed through SPAL 1. In this case, the SAF is close to 0, even if performance data across SPAL 1 seems to indicate otherwise. On the other hand, the SAF of moving Prefix p to SPAL 2 is, by contrast, very good, since the total load on the link will remain around 75% of total capacity, so delays will remain low. If, instead of carrying a load of 75% capacity, Prefix p carried a load of 10% capacity, the results would have been different, and the SAF of Prefix p across SPALs 1 and 2 would have been close. In some embodiments of the invention, without knowing the load of a link, we can still measure the effect of moving a given prefix to a given SPAL through RTT measurements. That is, instead of measuring the load directly, we measure the end result, that is the amount by which performance of prefixes across a link worsens as a result of moving a prefix to it.

Modifying the Schema for the Support of SAF

In order to support SAF, the schema may be include a load field in the SPAL table, and an SAF field in the PREFIX_SPAL table. In some embodiments, the SAF field is a per-prefix, per-SPAL information.

H. Available Bandwidth

Edge-stats measurements may include measurements of delay, jitter, and loss; using these measurements, an application-specific performance score may be obtained based on which a decision is made on whether to send an update request for this prefix. Available bandwidth is a valuable quantity that is measured and included in the computation of the performance score in some embodiments of the invention.

I. Differentiated Queues and Token Rates per Link

In some embodiments of the invention, token rates may differ on a per-link basis (which dictates the use of different queues for each link).

In some embodiments, the token rate may be tailored to total utilization. Lowly utilized links can afford relatively higher token rates without fear of overflow, whereas links close to saturation should be handled more carefully. Some embodiments of the invention provide one or more of the following modes of operation:

1. The default mode: the user specifies one token rate (and, optionally, a bucket size), shared equally among the prefixes updates destined to the different links.
2. The enhanced performance mode: the user specifies a minimum token rate (and, optionally, a bucket size). Depending on factors such as the total bandwidth

utilization and the bandwidth of individual links, the prefix scheduler takes the initiative to function at a higher speed when possible, allowing better performance when it is not dangerous to do so.

3. The custom mode: in this case, the user can specify minimum and maximum token rates (and, optionally, bucket sizes), as well as conditions on when to move from a token rate to another. Using this custom mode, customers can tailor the prefix scheduler to their exact need.

J. Prefix Winner set *Re-computation*

Even though the priority queue is sized in such a way that the delay spent in the queue is minimized, there is still an order of magnitude between the time scale of the BGP world, at which level decisions are taken, and the physical world, in which edge stats and interface stats are measured. That is, even though the queuing delay is comparable to other delays involved in the process of changing a route, prefix performance across a given link or the utilization of a given link can change much more quickly. For example, a 2 second queuing delay could be appropriate in the BGP world, while 2 seconds can be enough for congestion to occur across a given link, or for the link utilization to go from 25% to 75%... For this reason, in some embodiments of the invention, the winner set is re-evaluated at the output of the priority queue.

K. Conclusion

The foregoing description of various embodiments of the invention has been presented for purposes of illustration and description. It is not intended to limit the

invention to the precise forms disclosed. Many modifications and equivalent arrangements will be apparent.